UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

ANÁLISIS DE PATRONES EN DEPORTISTAS DE RESISTENCIA

Autor: Tomás Higuera Viso

Tutor: Gonzalo Martínez Muñoz

JULIO 2021

A mi familia y amigos…

Tomás Higuera Viso
Julio 2021

Resumen

El desarollo de la tecnología aplicada al deporte avanza a diario. Cada día son más los entrenadores y deportistas que la utilizan para mejor la calidad de sus entrenamientos y aumentar su rendimiento.

Este Trabajo de Fin de Grado se ha centrado en utilizar los datos que generan los deportistas de resistencia en sus entrenamientos para detectar patrones que nos permitan predecir lesiones o puntos óptimos de forma haciendo uso del aprendizaje automático.

Al comienzo de este trabajo no se disponía de ninguna base de datos de entrenamientos, por lo que para poder lograr nuestro objetivo ha sido necesario analizar las posibles fuentes para obtener estos datos. Finalmente se ha optado por utilizar Garmin Connect como fuente de estos datos, ya que permite descargar masivamente el histórico de actividades de cada deportista para su posterior análisis.

Una vez elegida la fuente de datos se ha procedido a llenar la base de datos con entrenamientos de deportistas de resistencia. Para esta tarea ha sido necesario crear una página web junto con un cuestionario en la que el deportista ha accedido a prestar sus datos de entrenamientos para el estudio. La página web ha sido desplegada en *Heroku*(Plataforma de servicios de computación en la nube) y se ha utilizado el *framekork* de Python, *Django*.

Una vez obtenidos los datos ha sido necesario procesarlos y filtrarlos, ya que los archivos de entrenamiento que nos proporcionaba Garmin Connect estaban en formato GPX (esquema XML para transferir datos GPS), por lo que tuvimos que transformarlos para poder utilizarlos en los modelos de aprendizaje automático. El objetivo final de esta transformación era obtener los TSS(*Training Stress Score*) de cada entrenamiento. Esta unidad de medida se utiliza en la aplicación deportiva TrainingPeaks y refleja la fatiga de un deportista en cada entrenamiento.

Posteriormente se han visualizado los datos junto con las respuestas que habían dado los deportistas en el cuestionario y mediante el análisis de las gráficas obtenidas, se han extraído conclusiones acerca de los patrones que se producían en la vida deportiva de cada individuo. Con estas conclusiones se han obtenido los punto característicos que han sido utilizados posteriormente para realizar predicciones a partir de un modelo de aprendizaje.

Palabras clave: Predicción, Python, Django, MVC, MVT, web framework, template, regresión lineal, aprendizaje automático, TSS, ATL, CTL, TSB, ritmo normalizado, umbral de potencia funcional, factor de intensidad, GPX.

Abstract

The development of technology applied to sport grows daily. Every day there are more coaches and athletes that uses technology as a way to improve their performance and the quality of their training sessions.

This Final Degree Project is focused on use the data generated by athletes in their training sessions to detect patterns which help us to detect injuries or optimal shape points making use of automatic learning.

At the beginning of this work, we did not have any training sessions database, so to achieve our goal we had to analyze the possible sources to obtain this data. Finally we have chosen Garmin Connect, because it let us download massively all the activities from each athlete for their further analysis.

Once we chose our source we proceed to fill the database with the training sessions of resistance athletes. For this goal it was necessary to create a web page with a questionnaire in which the athlete had agreed to provide his training data for the study. The web page have been deployed in *Heroku*(Cloud platform) using the Python *framekork* Django.

Once we obtained the data it was necessary to process and filtrate it, because the training sessions that Garmin Connect provide us were in GPX format (XML schema for GPS data transfer), so we had to transform them so we could use the in machine learning models. The aim of this transformation was to obtain the TSS(*Training Stress Score*) of each session. This measurement is used in the sport application TrainingPeaks and mirrors in a really precise way the shape of an athlete.

Later we visualized the data along with the answers given by the athletes in the questionnaire and by using the different graphs, we have extracted conclusions about the patterns that get produced in the sport life of each individual. With this conclusions we have extracted characteristics points that we used for making predictions from a learning model.

Key words: Prediction, Python, Django, MVC, MVT, web framework, template, linear regression, automatic learning, TSS, ATL, CTL, TSB, normalized pace, functional threshold power, intensity factor, GPX.

Agradecimientos

En primer lugar, me gustaría dar las gracias a mi tutor Gonzalo Martínez Muñoz por darme la oportunidad de realizar este trabajo, desde que se lo propuse ha estado dispuesto a ayudarme y a darme consejos siempre que lo necesitaba.

En segundo lugar, me gustaría dar las gracias a Alejandro Arias Diez, Oscar Fernández Sánchez y Jaime Gil Cabrera, por las charlas sobre planificación de entrenamientos y teoría del deporte, sin su ayuda este trabajo no habría salido adelante.

Agradecer también a mi club, Perlas Triatlon Colmenar Viejo, por haberme prestado los datos con los que trabajar, una parte fundamental de este trabajo, ya que no disponíamos de ninguna base de datos.

También me gustaría agradecer a mis amigos, por estar ahí para echarme una mano cuando lo necesitaba y por todas las experiencias vividas durante estos años de carrera.

Por último, me gustaría agradecer mis padres y mi hermana, por el apoyo incondicional durante estos años.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Este proyecto se encuentra motivado en gran medida por el creciente interés por los deportes de resistencia en nuestra sociedad [1]. Cada vez más gente practica deporte para mantener un modo de vida saludable o tratar de batir sus marcas.

El hecho de que cada vez más gente practique deporte ha llevado a la tecnificación de los mismos. Esta viene dada por la utilización de nuevas tecnologías y la aplicación de las mismas con el objetivo común de que deportistas profesionales y amateur puedan mejorar su rendimiento de manera óptima sin dejar de lado las posibles lesiones que se pueden producir por una mala gestión de los entrenamientos.

La tecnología ha calado de lleno en los deportes de resistencia con el uso de relojes inteligentes, potenciómetros, sensores de cadencia, pulsómetros, etc. Todos estos datos son utilizados por entrenadores y deportistas como referencia para poder conocer la carga de entrenamientos o estado de forma y en ocasiones, pueden dejar de ser de ayuda para los que los utilizan, ya que son demasiado densos y por tanto, complicados de analizar con una perspectiva global. Conociendo esta situación, la motivación de este proyecto es obtener una visión global de la evolución de un deportista de resistencia a lo largo de su carrera deportiva, para predecir picos de forma o en función de las últimas cargas de entrenamiento percibidas, predecir lesiones por sobrecarga de entrenamientos. El objetivo final de deste proyecto es dotar a entrenadores y deportistas de una herramienta que contribuya a la tecnificación de este tipo de deportes.

1.2. Objetivos

Con todo lo expuesto en la sección anterior, los objetivos que se han definido en este proyecto son los siguientes:

- **Estudio de métodos de entrenamiento:** Antes de afrontar este proyecto es necesario conocer las diferentes técnicas que utilizan los entrenadores para hacer planes

de entrenamiento y en que se basan para establecer las cargas de los mismos.

- **Recopilación de datos:** Es necesario obtener una gran cantidad datos de deportistas de cualquier nivel a través de plataformas deportivas, para poder tener una visión general de como los entrenamientos afectan a cada tipo de persona.
- **Procesamiento de datos:** Una vez obtenidos los datos es necesario analizarlos, procurando establecer un margen para limpiar los datos atípicos que puedan impedir que el proyecto se realice correctamente.
- **Visualización de datos:** La mejor manera de buscar patrones en los entrenamientos de cada individuo es visualizar los datos obtenidos, para poder establecer los valores que se analizarán en los modelos de aprendizaje automático.
- **Modelos de predición:** Crear modelos de predección para detectar picos de forma de un deportista o predecir lesiones por sobrecarga de entrenamiento.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** Introducción.
- **Capítulo 2:** Estado del arte.
- **Capítulo 3:** Desarrollo.
- **Capítulo 4:** Base de datos.
- **Capítulo 5:** Pruebas y resultados.
- **Capítulo 6:** Conclusiones y trabajo futuro.

Al final del documento se incluyen los anexos que añaden y complementan la información expuesta a lo largo de esta memoria.

Capítulo 2

Estado del arte

En este capítulo se desarrollará la base teórica sobre la que se asienta este trabajo. En primer lugar se explicará el web framework que se ha utilizado para este proyecto y la plataforma en la que se ha alojado el mismo. A continuación se explicarán algunos conceptos generales sobre el aprendizaje automático, haciendo hincapié en los algoritmos utilizados en este proyecto, la regresión lineal y random forest. Y por último se describirán las aplicaciones deportivas utilizadas en la actualidad, concretamente dos, Garmin Connect y TrainingPeaks, así como los diferentes métodos que utilizan para la extracción y análisis de datos.

2.1. Web framework

El *web framework* que se ha utilizado para el desarrollo web ha sido **Django** [2]. Este *web framework* utiliza como lenguaje de desarrollo Python. **Django** utiliza una versión redefenida del MVC (Modelo Vista Controlador) [3] el MVT (Modelo Vista Template). Este nuevo patrón es muy similar al MVC con la diferencia que en el caso de Django se recibe una petición y después de realizarse las acciones en la vista, se renderiza el template.

La razón de elegir **Django** *web framework* es la versatilidad que da a la hora de añadir módulos como Bootstrap [4] para el frontend, Postgresql [5] para la base de datos o la integración con servicios de computación en la nube como Heroku [6]. Otra de los motivos de nuestra elección ha sido la estabilidad de este *web framework*.**Django** nació en 2003, cuando los programadores web Adrian Holovaty y Simon Willison comenzaron a usar Python para el desarrollo de aplicaciones. Este *web framework* ha continuando desarrollándose a lo largo del tiempo, lo que lo hace muy atractivo para la comunidad. Tanto es así, que **Django** paso de 40 millones de descargas en 2019 a 60 millones en 2020, lo que supone un 50 % de crecimiento.

2.2. Servicios de computación en la nube

Lo servicios de computación en la nube [7] nacen de la necesidad de poder alojar servicios de computación en internet. La aparición de la computación en la nube supuso un gran avance en la informática ya que permite tener distribuidos servicios como bases de datos o servidores web en cualquier parte del mundo. Las ventajas que esto supone son muchas, a continuación se describen algunas:

- **Disponibilidad**: Tener acceso a múltiples servidores a lo largo del mundo permite, entre otras cosas, que si en algún momento uno de los servidores falla, se puede recuperar fácilmente el servicio redirigiendo la carga a otra servidor en otro lugar.
- **Escalabilidad horizontal**: El hecho de poder tener diferentes servidores alojados en diferentes lugares permite repartir la carga y poder tener mayor potencia computacional.
- **Globalización**: Otra de las ventajas de la computación en la nube es que pemite alojar los servicios que ofreces en cualquier parte del mundo, lo que abre la posibilidad a cualquier persona de desarrollar su servicio en su ciudad natal y poder ofrecérselo a cualquier otra persona del mundo simplemente utilizando servidores más cercanos a ella para mejorar el rendimiento.

Estas son algunas de las características que hacen de los servicios de computación en la nube una tecnología que tener en cuenta en la actualidad. Hay infinidad de posibilidades a la hora de elegir una plataforma que ofrezca estos servicios para alojar servidores web (Microsoft Azure, AWS, Google Cloud, etc). En nuestro caso hemos elegido **Heroku** para almacenar nuestro servidor web. La razón principal de haber elegido esta plataforma es la facilidad de la misma para integrar una aplicación creada con el *framework* Django. Las otras opciones que se nos ofrecían en el mercado, a pesar de ser mucho mejores en cuanto a potencia computacional, eran demasiado para el enfoque que tenía nuestro proyecto, es decir, poder alojar un servidor web que no sea estático y pueda estar asociado a una base de datos escalable, ya que necesitábamos una gran cantidad de almacenamiento para poder almacenar todas las actividades de los deportistas. Otra de las características que hizo que eligiéramos **Heroku** como nuestro servicio de computación en la nube es su bajo precio. Los servidores web alojados en esta plataforma pasan a un estado de hibernación cuando están un tiempo sin utilizarse, lo que abarata en gran medida los costes para un proyecto como el nuestro. Además **Heroku** permite descargar backups de la base de datos muy fácilmente y de manera gratuita, una funcionalidad requerida en este proyecto ya que necesitábamos trabajar con todos los datos obtenidos de manera local.

2.3. Aprendizaje automático

Lo primero que debemos explicar antes de entrar de lleno en el proyecto es el tipo de problema que queremos abordar y que tipo de tecnologías usar para resolverlo. El aprendizaje automático [8] es una rama de la Inteligencia Artificial enfocada a crear modelos de patrones para poder predecir comportamientos y clasificar datos. Dependiendo de si el método de aprendizaje automático conoce o no la solución al problema, podemos agrupar los algoritmos en:

- **Aprendizaje supervisado**: En este tipo de algoritmos, durante la fase de entrenamiento el modelo conoce tanto los valores de entrada como los resultados deseados. Existen multitud de algoritmos que se basan en este tipo de aprendizaje, podemos destacar vécinos próximos, regresión logistica, árboles de decisión, etc. La elección del algoritmo a utilizar dependerá finalmente de las características del problema a resolver.
- **Aprendizaje no supervisado**: Este tipo de algoritmos se diferencia del anterior en que en la fase de entrenamiento el modelo solo conoce los valores de entrada, por lo que no tiene un conocimiento a priori de problema. El algoritmo detecta patrones en los datos de entrada que producen determinadas salidas y agrupa los datos en consecuencia. Algunos ejemplos de este tipo de algoritmo son las redes neuronales de Kohonen o los algoritmos de clustering.

Regresión lineal

La regresión lineal es un modelo matemático utilizado para calcular una variable dependiente en función de los valores de las variables independientes. Las variables independientes son las características del problema, es decir, los valores con los que se realiza la predicción y la variable dependiente es el objetivo, la variable que se intenta predecir. La ecuacuón general de los modelos de regresión es la siguiente:

$$Y = \beta_0 + \sum_{i=1}^{N} \beta_i X_i + \epsilon_i$$

- β: Son las variables independientes.
- ϵ: Son los términos de error.
- N: Es el número de variables independientes

El caso más sencillo de regresión lineal se produce cuando solo tenemos un parámetro independiente con lo que el resultado obtenido es una recta. Un ejemplo de recta regresión se puede observar en la Figura 2.1

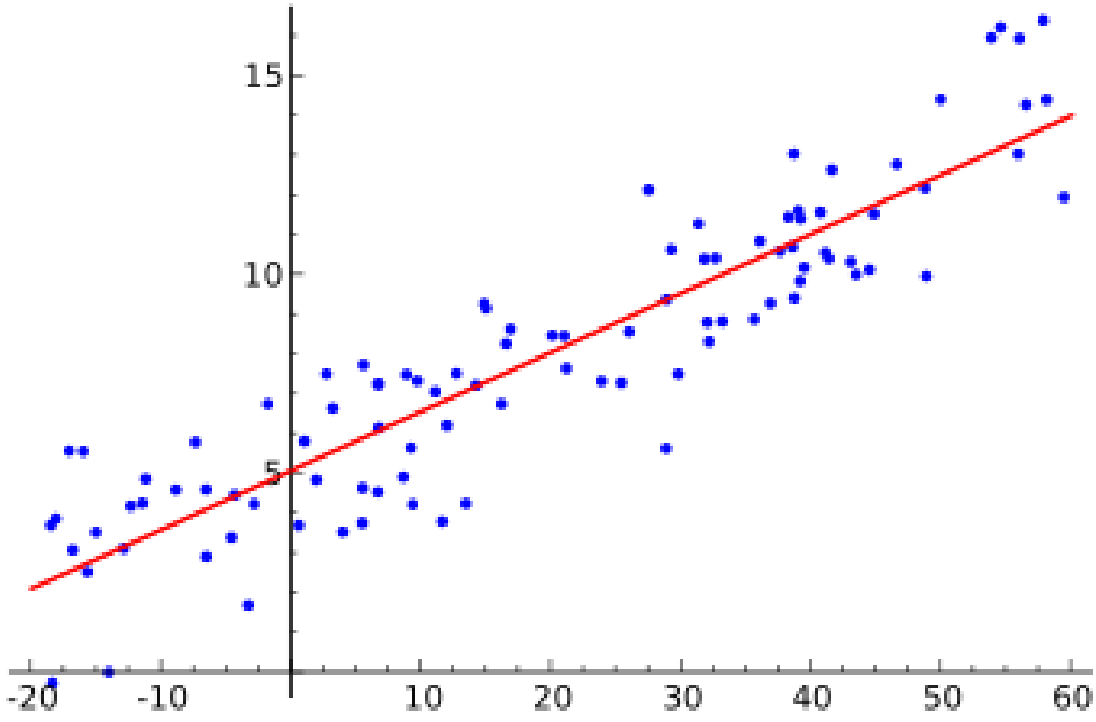

FIGURA 2.1: Recta de regresión.

Random forest

Random forest es un algoritmo de aprendizaje automático de tipo supervisado. Este algoritmo es un conjunto de árboles de decisión que procesan las variables independientes del problema hasta dar con un valor de salida para la variable dependiente. La salida de Random Forest es una regresión promediada de las salidas de los árboles para poder obtener una predicción más precisa, como se puede observar en la Figura 2.2.

La construcción de los bosques aleatorios sigue la siguiente metodología: en cada nodo del arból se toma de manera aleatoria una cantidad determinada de variables explicativas y se construye un árbol. Cuando se tienen todos los árboles se promedia la salida y con ello obtenemos la decisión final del algoritmo.

2.4. Aplicaciones deportivas

Hoy en día hay infinidad de dispositivos con los que realizar mediciones de datos en los entrenamientos. Estos dispositivos pueden ser potenciómetros, relojes con GPS, cadenciómetros, etc. Existen multitud de marcas que producen este tipo de dispositivos y con cada marca, una aplicación diferente que se encarga de recoger los datos.

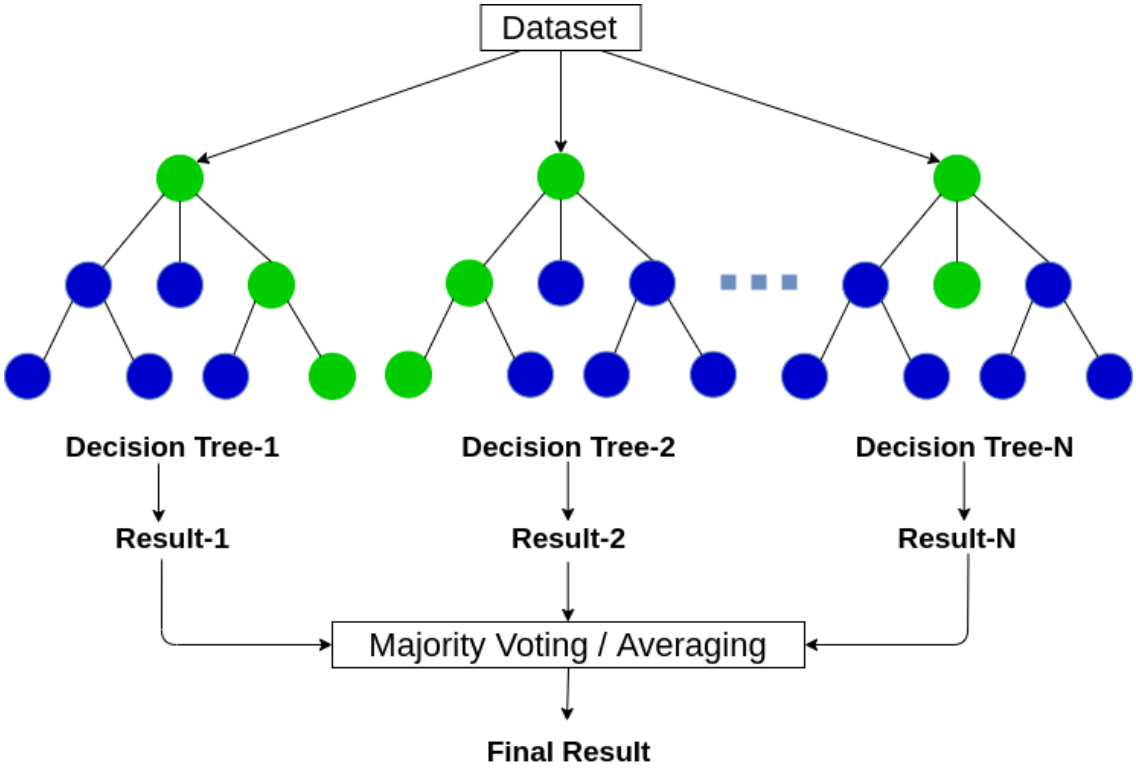

FIGURA 2.2: Algoritmo Random Forest. Imagen extraída de [9].

2.4.1. Garmin

Para este proyecto necesitábamos una fuente de la que obtener los datos de los entrenamientos de deportistas. Teníamos multitud de opciones, Polar, Garmin, Suunto, etc. Finalmente optamos por elegir Garmin, ya que es una de las marcas más punteras en este tipo de dispositivos y cuenta con una aplicación llamada Garmin Connect[10] que agrupa todas las actividades de los deportistas que hacen uso de sus dispositivos. Además, esta aplicación cuenta con un módulo de Python que nos permitirá conectárnos a los servidores de Garmin Connect y descargar masivamente todas las actividades de los deportistas.

2.4.2. TrainingPeaks

TrainingPeaks [11] es una aplicación puntera en la planificación de entrenamientos, ya que cuenta con una unidad de medida desarrollada por ellos, llamada TSS (Trainning Stress Score) [12]. Esta unidad de medidad sirve para medir la fatiga que soportan los deportistas en cada entrenamiento. Con esta medida se pueden obtener otros valores como Fitness, Fatigue y Form, que permiten a los entrenadores obtener una mejor referencia de cómo sus deportistas están asimilando la carga de entrenamientos. Además esta aplicación sirve para recopilar los datos de dispositivos de distintas marcas y poder visualizarlos de una infinidad de maneras.

Trás hablar con algunos entrenadores, decidimos utilizar esta unidad como referencia

del esfuerzo de los deportistas en cada entrenamiento para los modelos de aprendizaje automático que ibamos a implementar.

Capítulo 3

Desarrollo

En este capítulo se explicarán las distintas decisiones que se han tomado en la fase de desarrollo del proyecto. En concreto se explicará cómo se han obtenido y los datos y cómo han sido procesados y filtrados para la limpieza de datos atípicos. Finalmente se mostrarán algunas gráficas y explicaciones de los datos obtenidos.

3.1. Obtención de los datos

En esta sección se detallará la fuente de la que se han obtenido los datos, y las diferentes APIs utilizadas. También se explicará la estructura de los datos obtenidos y la necesidad de procesarlos y filtrarlos para obtener los datos requeridos.

3.1.1. Garmin Connect

Para la obtención de los datos actividades hemos hecho uso de la aplicación de Garmin, Garmin Connect. Esta aplicación, a pesar de contar con API, no está enfocada a desarrolladores, es por eso, que hemos tenido algunos problemas a la hora de obtener los datos.

Al comienzo del desarrollo empezamos utilizando el módulo de Python *garminconnect*. Este módulo nos permitía iniciar sesión en Garmin y descargar las actividades cuando proporcionábamos un *Id* de actividad.

```
1  from garminconnect import (
2      Garmin,
3      GarminConnectConnectionError,
4      GarminConnectTooManyRequestsError,
5      GarminConnectAuthenticationError,
6  )
7  # Inicio de sesión
8  cliente = Garmin(username, password)
9  cliente.login()
10
```

```
# Obtención de ids de actividades
actividades = client.get_activities(0,1)  # 0=start, 1=limit
# Descarga de actividades
for actividad in actividades:
    id_actividad = actividad["activityId"]
    gpx_data = client.download_activity(id_actividad, dl_fmt=
        client.ActivityDownloadFormat.GPX)
```

El problema que nos encontrábamos con esta matodología para descargar las actividades masivamente es que no podíamos obtener los *Ids* de actividad de manera sencilla ya que, la función *get_activities* solicitaba como argumento un índice de inicio y un índice de fin, datos que no conocíamos de los usuarios, por lo que si queríamos hacer una descarga masiva de los datos de cada deportista, nos iba a ser muy difícil. En este momento, se nos ocurrió desarrollar un *Web Crawler* para la página web de la aplicación Garmin Connect, que navegará por la web y descargará las actividades una a una. Pero nos volvíamos a encontrar otra vez con el problema de la descarga masiva de datos. Este método no era funcional ya que, para acceder a la página web de Garmin Connect se necesitaba un usuario y una contraseña, y si queríamos automatizar el proceso de obtención de datos no queríamos tener que almacenar estos datos sensibles. Finalmente encontramos otro módulo llamado *garminexport* [13], que a pesar de estar obsoleto, aún podíamos descargar y utilizar en nuestro proyecto. Este módulo lo usaríamos para obtener un listado de todos los *Ids* de actividades de cada deportista, el resto del proceso de descarga lo realizaríamos con el otro módulo *garminconnect*.

```
from garminconnect import (
    Garmin,
    GarminConnectConnectionError,
    GarminConnectTooManyRequestsError,
    GarminConnectAuthenticationError,
)
from garminexport.garminclient import GarminClient as
    GarminExport
# Inicio de sesión
cliente = Garmin(username, password)
clienteExport = GarminExport(username, password)
cliente.login()
clienteExport.connect()

# Obtención de ids de actividades
actividades = clienteExport.list_activities()  # 0=start, 1=limit
# Descarga de actividades
for actividad in actividades:
    id_actividad = actividad[0]
    gpx_data = client.download_activity(id_actividad, dl_fmt=
        client.ActivityDownloadFormat.GPX)
```

3.1.2. Archivos GPX

Una vez resuelto el problema de la descarga masiva de actividades, el siguiente punto a tratar era la metodología que utilizaremos para manipular los archivos GPX. Los

archivos GPX son ficheros utilizados para el intercambio de mensajes entre GPS. Estos archivos siguen un esquema XML pensado para la transferencia de coordenadas, con el objetivo de transferir puntos y recorridos en un mapa. Cada uno de estos ficheros tiene un listado de capturas realizadas por el GPS con información acerca de las coordenadas, altitud, instante de tiempo en el que se realizo la captura y frecuencia cardíaca (si el deportista contaba con banda de frecuencia en el entrenamiento). Para parsear estos datos a una estructura con la que poder trabajar hemos hecho uso del módulo de Python *xml* [14], que nos permite transformar un fichero con estructura XML en una estructura tipo Document, mucho mas accesible para trabajar con los datos. En la Figura 3.1 se muestra un ejemplo del formato de este tipo de archivos.

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx creator="Garmin Connect" version="1.1"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX/11.xsd"
  xmlns:ns3="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
  xmlns="http://www.topografix.com/GPX/1/1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns2="http://www.garmin.com/xmlschemas/GpxExtensions/v3">
  <metadata>
    <link href="connect.garmin.com">
      <text>Garmin Connect</text>
    </link>
    <time>2016-11-20T09:37:06.000Z</time>
  </metadata>
  <trk>
    <name>Presa De Hidraulica (Santillana) Carrera</name>
    <type>running</type>
    <trkseg>
      <trkpt lat="40.654478659853339195251464843 75" lon="-3.765093991532921791076660156 25">
        <ele>849</ele>
        <time>2016-11-20T09:37:06.000Z</time>
        <extensions>
          <ns3:TrackPointExtension/>
        </extensions>
      </trkpt>
      <trkpt lat="40.654456783086061477661132812 5" lon="-3.765098853036761283874511718 75">
        <ele>849</ele>
        <time>2016-11-20T09:37:07.000Z</time>
        <extensions>
          <ns3:TrackPointExtension/>
        </extensions>
      </trkpt>
      .
      .
      .
      .
    </trkseg>
  </trk>
</gpx>
```

FIGURA 3.1: Formato de un archivo GPX.

3.2. Formateo y limpieza de los datos

En esta sección se detallará los siguientes pasos a seguir después de obtener los datos de Garminconnect. En este momento del proyecto disponemos de ficheros GPX con el resumen de una sesión de entrenamiento. Para que podamos utilizar este tipo de ficheros es necesario formatear los datos, ya que la gran mayoría de las fórmulas aplicadas a entrenamientos no trabajan con coordenadas, sino con velocidad. Además debemos elegir una estructura para almacenar cada actividad y una estructura que nos sirva para almacenar los datos de cada deportista.

3.2.1. Almacenamiento de actividades

Lo primero que necesitamos es obtener la velocidad en los entrenamientos. Para ello hemos aplicado la fórmula de Haversine [15], que sirve para hallar la distancia entre dos puntos geográficos. Una vez obtenida la distancia, la dividimos entre el tiempo entre ambas capturas del GPS y obtenemos la velocidad.

Una vez obtenida la velocidad hemos elegido el tipo de estructura en la que almacenaremos los datos de cada actividad. La estructura elegida es un diccionario de Python. Este diccionario debe contener información relativa a la fecha, *Id* de actividad, velocidad, desnivel, etc. Los campos del diccionario serán los siguientes:

- **activity_id**: Contiene el *Id* del entrenamiento.
- **type**: Contiene el tipo de actividad (natación, carrera, ciclismo)
- **date**: Contiene la fecha de la actividad.
- **time**: Contiene una lista con los instantes de tiempo en los que se realizaron las capturas del GPS.
- **total_time**: Contiene el tiempo total que ha transcurrido en la actividad.
- **distance**: Contiene una lista con las distancias recorridas entra cada captura del GPS.
- **total_distance**: Contiene la distancia total recorrida en la actividad.
- **pace**: Contiene una lista con las velocidades entre cada captura del GPS.
- **normalized_pace**: Contiene una lista con las velocidades normalizadas en función de la pendiente entre cada captura del GPS.
- **hearthrate**: Contiene una lista con las frecuencias cardiacas en cada captura del GPS.
- **elevation**: Contiene una lista con las altitudes en cada captura del GPS.
- **elevation_grade**: Contiene una lista con los porcentajes de inclinación entre cada captura del GPS.

3.2.2. Almacenamiento de datos relativos a un deportista

Una vez definida la estructura de cada actividad es necesario abordar el modo en que almacenaremos los datos de cada deportista. Para ello hemos creado una clase *DataAnalyzer* que contiene todas las actividades del deportista. Los datos que almacenaremos de cada deportista son los siguientes:

- **username**: Contiene un nombre que utilizaremos para identificar al deportista.
- **age**: Contiene la edad del deportista.
- **max_hearthrate**: Contiene la frecuencia cardiaca máxima que ha registrado el deportista entre todas sus actividades.
- **calendar**: Contiene un diccionario con un objeto de tipo *datetime* como clave, que contiene una lista de las actividades realizadas en ese día.
- **data**: Contiene un diccionario con los tipos de deporte como clave, que contiene un listado de actividades relativas a cada tipo.

3.3. Procesamiento de los datos

Después de almacenar los datos de actividades relativos a cada deportista, pasamos a investigar la manera en la que los procesamos para obtener una perspectiva objetiva de la intensidad de cada entrenamiento. Tras hablar con múltiples entrenadores, decidimos optar por los TSS (Training Stress Score), una medida utilizada para medir la intensidad de los entrenamientos.

3.3.1. Ritmo normalizado

Para calcular los TSS de un entrenamiento es necesario calcular primero el ritmo normalizado de la actividad. La necesidad de este cálculo es que la velocidad en un entrenamiento varía en función de los desniveles encontrados en la actividad, es decir, en desniveles positivos el ritmo disminuye, y en desniveles negativos el ritmo se incrementa. Estos cambios de ritmo hacen que no tengamos una referencia real del ritmo del deportista en la actividad, por lo que es necesario que normalicemos la velocidad en función del porcentaje de desnivel entre cada punto capturado por el GPS. Para esta normalización, la gran mayoría de las aplicaciones deportivas hacen cálculos en función de los datos que tienen almacenados y realizan una estimación de este ritmo. El problema que nos encontramos es que nosotros no disponemos de tal volumen de datos, por lo que no podemos seguir la misma metodología. Trás una búsqueda exhaustiva encontramos una tabla que contenía la equivalencia de ritmos en función del porcentaje de desnivel, esta se muestra en la Tabla 3.1 [16]. Con esta tabla realizamos una regresión lineal que estimaría el ritmo normalizado dado un ritmo y una pendiente.

Una vez obtenidos los ritmos normalizados pasamos a visualizar una comparativa contra el ritmo normal haciendo uso del módulo matplotlib de Python. En la Figura 3.2 se muestra una comparativa tras realizar este procesamiento.

TABLA 3.1: Tabla de ritmos normalizados en función de la pendiente.

Velocidad (mph)	**Porcentaje de inclinacion**									
-	**1 %**	**2 %**	**3 %**	**4 %**	**5 %**	**6 %**	**7 %**	**8 %**	**9 %**	**10 %**
12:00	12:31	11:44	11:05	10:32	10:03	9:38	9:16	8:56	8:38	8:22
11:32	12:02	11:18	10:42	10:11	9:44	9:20	8:59	8:40	8:23	8:08
11:07	11:35	10:55	10:20	9:51	9:26	9:03	8:43	8:25	8:09	7:55
10:43	11:10	10:32	10:00	9:33	9:09	8:48	8:29	8:12	7:56	7:42
10:21	10:47	10:12	9:42	9:16	8:53	8:33	8:15	7:58	7:44	7:30
10:00	10:26	9:52	9:24	9:00	8:38	8:19	8:02	7:46	7:32	7:19
9:50	10:15	9:43	9:16	8:52	8:31	8:12	7:55	7:40	7:26	7:14
9:41	10:05	9:34	9:08	8:44	8:24	8:06	7:49	7:34	7:21	7:08
9:31	9:56	9:26	9:00	8:37	8:17	7:59	7:43	7:29	7:15	7:03
9:23	9:46	9:17	8:52	8:30	8:10	7:53	7:37	7:23	7:10	6:58
9:14	9:37	9:09	8:45	8:23	8:04	7:47	7:32	7:18	7:05	6:53
9:05	9:29	9:01	8:37	8:16	7:58	7:41	7:26	7:13	7:00	6:49
8:57	9:20	8:53	8:30	8:10	7:52	7:35	7:21	7:07	6:55	6:44
8:49	9:12	8:45	8:23	8:03	7:46	7:30	7:15	7:02	6:50	6:40
8:42	9:04	8:39	8:17	7:57	7:40	7:24	7:10	6:58	6:46	6:35
8:34	8:53	8:32	8:10	7:51	7:34	7:19	7:05	6:53	6:41	6:31
8:27	8:45	8:25	8:04	7:45	7:29	7:14	7:00	6:48	6:37	6:27
8:20	8:41	8:18	7:58	7:40	7:23	7:09	6:56	6:44	6:33	6:22
8:13	8:34	8:12	7:52	7:34	7:18	7:04	6:51	6:39	6:28	6:18
8:06	8:27	8:05	7:46	7:28	7:13	6:59	6:46	6:35	6:24	6:14
8:00	8:20	7:59	7:40	7:23	7:08	6:54	6:42	6:31	6:20	6:11
7:54	8:14	7:53	7:34	7:18	7:03	6:50	6:38	6:26	6:16	6:07
7:48	8:07	7:47	7:29	7:13	6:58	6:45	6:33	6:22	6:12	6:03

3.3.2. TSS

Una vez obtenidos los ritmos normalizados pasamos a calcular los TSS [12]. Los TSS son una medida utilizada para medir la intensidad de un entrenamiento. La forma de calcularlos es la siguiente:

$$TSS = [(NP * IF * S)/(FTP * 3600)] * 100$$

- TSS (Training Stress Score): Puntuación de intensidad del entrenamiento.
- S (Seconds): Duración del entrenamiento en segundos.
- NP (Normalized Power): Ritmo normalizado en función de la pendiente durante el entrenamiento.
- FTP (Functional Threshold Power): Ritmo umbral funcional del deportista.
- IF (Intensity Factor): Factor de intensidad que se calcula con la siguiente fórmula: $IF = NP/FTP$.

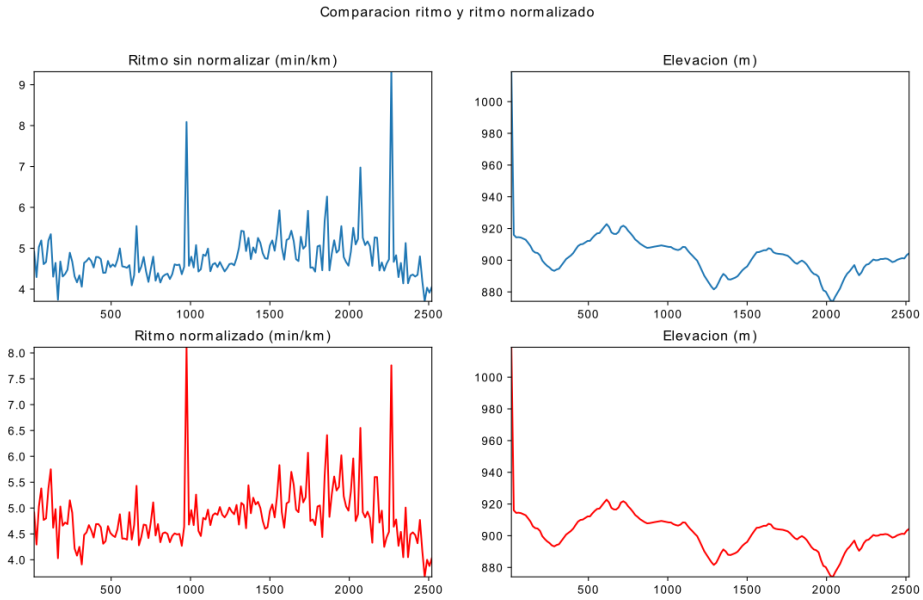

FIGURA 3.2: Comparacion ritmo normalizado y sin normalizar.

Como podemos observar en la fórmula, aún hay un dato que no conocemos, el FTP. El FTP es el ritmo más alto que un atleta puede mantener en una sesión de veinte minutos. Este valor varía en función del estado físico del deportista. Normalmente, para calcular este valor, es necesario realizar una prueba de esfuerzo o un ejercicio específico en el que se obtiene este valor. Nuestro objetivo era obtener el FTP y como variaba el mismo a lo largo del tiempo. Para esta tarea analizamos intervalos de treinta días en el histórico de actividades del deportista y obtuvimos el ritmo más alto mantenido en veinte minutos. Este cálculo nos funcionaba en prácticamente todos los deportistas, ya que es normal realizar test de veinte minutos como entrenamiento en la preparación de una competición. El problema aparecía con los deportistas que trabajaban a menos intensidad y por tanto el mejor ritmo en veinte minutos no era una referencia. Para mejorar el cálculo decidimos tomar como referencia la frecuencia cardíaca durnate los entrenamientos. Lo primero que hicimos fue obtener la frecuencia cardiaca máxima registrada por el deportista. Una vez obtenida esa frecuencía y obtenido el mejor parcial de veinte minutos en una franja de treinta días, comprobamos la frecuencia cardíaca media que se mantuvo en el parcial, y junto a la frecuencia cardíaca máxima calculamos cuanto le quedaba al deportista para llegar al 85 % de esa frecuencia y con ello reajustar el FTP.

Con estos datos ya podíamos calcular los TSS del deportista, así como otras variables que dependían de estos, Fitness (CTL), Fatigue (ATL) y Form (TSB). El problema que nos encontrábamos ahora es que este cálculo requería de mucha potencia computacional para completarse, ya que guárdabamos una gran cantidad de datos en las actividades debido a que los GPS realizan muchas capturas de posición por segundo. La solución que optamos por implementar fue dividir la actividad en franjas de quince segundos, en las que calculamos velocidad media, distancia y porcentaje de desnivel. Este calculo redujo notablemente el tiempo de ejecución, 191.63 (tres minutos y veinte segundos) en

procesar un lote de 1246 actividades, como se puede observar en la Figura 3.3.

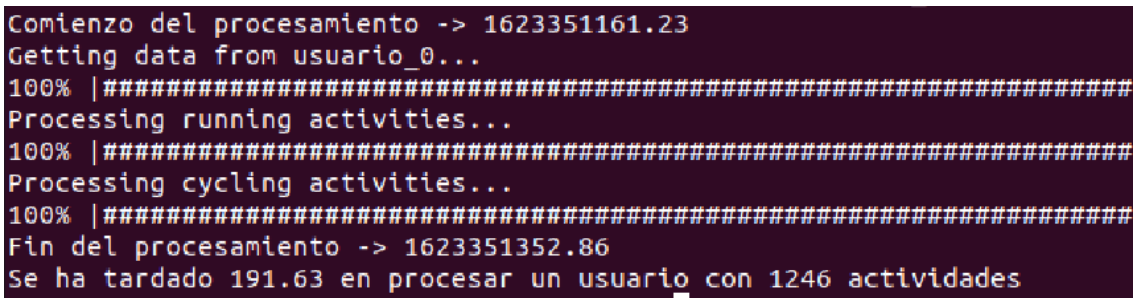

FIGURA 3.3: Tiempo total de procesamiento de un deportista.

3.4. Visualización de los datos

Con estos datos ya podíamos empezar a realizar analíticas de los patrones que se producían en el histórico de actividades de un deportista. Para esta visualización utilizaremos el módulo de Python matplotlib [17]. Aunque antes de comenzar con este análisis realizamos una comprobación de los datos que habíamos obtenidos para saber si nuestra aproximación fue la correcta. Para esta comparación obtuvimos los TSS de un deportista a lo largo de un año y lo contrastamos con los TSS calculados por la aplicación Training Peaks. En la Figura 3.4 se muestran los resultados obtenidos, los TSS de la parte superior son los obtenidos tras nuestro procesamiento y los de la parte inferior los de la aplicacion Training Peaks.

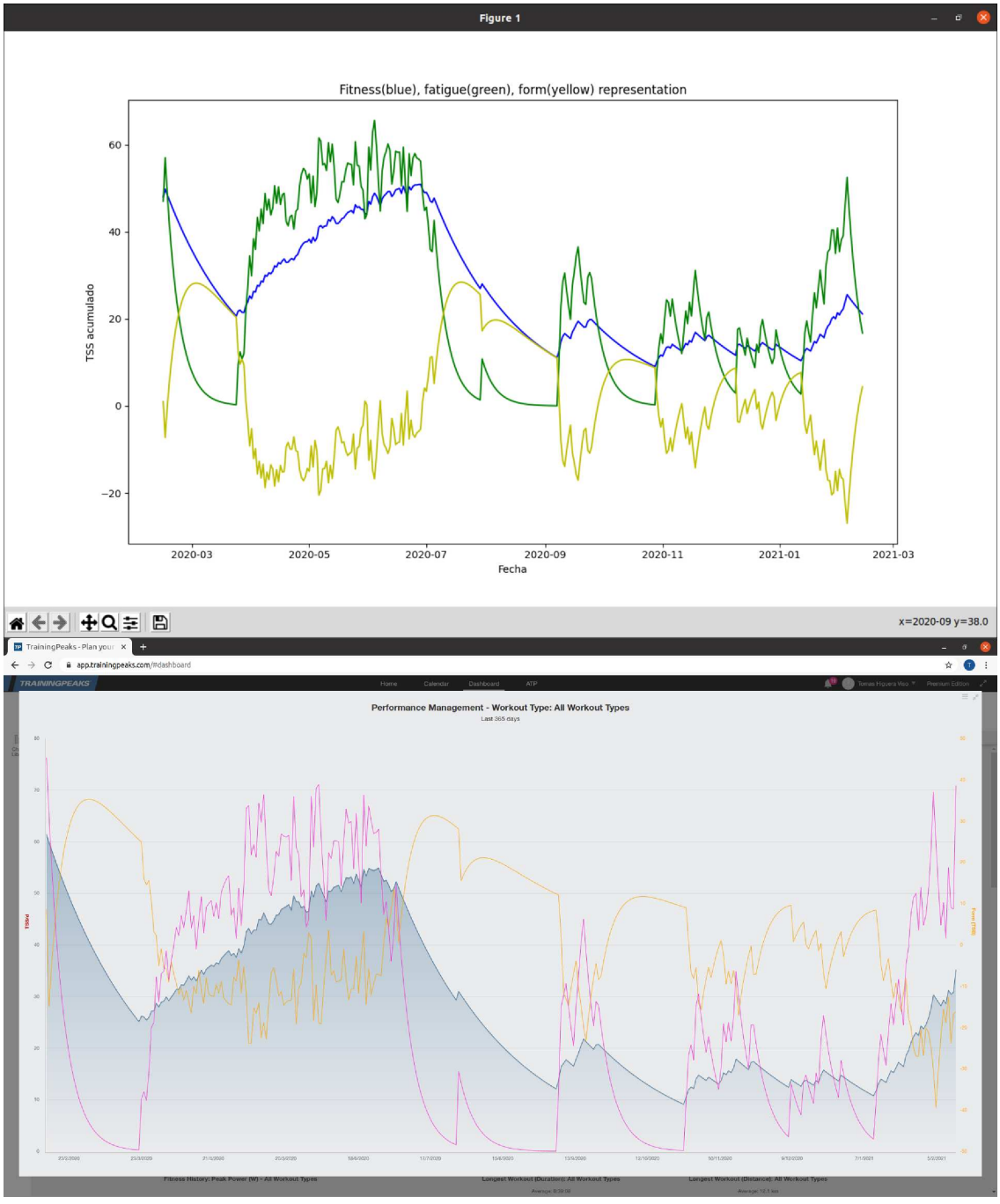

FIGURA 3.4: Comparacion TSS con Training Peaks.

Capítulo 4

Base de datos

En este capítulo se expondrán las distintas decisiones que se han tomado en la fase de creación de la base de datos para el proyecto. Primero se definirá la base de datos que se ha utilizado, seguiremos con la recopilación de los datos y por último la elección de patrones para aprendizaje automático.

4.1. Servidor web para la base de datos

En esta sección se explicarán los diferentes motivos por los que optamos por utilizar un servidor web para recopilar los datos que utilizaremos en el proyecto. Uno de los problemas a la hora de afrontar este proyecto fue que no disponíamos de base de datos de entrenamientos. Al querer aplicar algoritmos de aprendizaje automático para la detección de patrones, era necesario disponer de una con la mayor cantidad de datos posibles. Como hemos explicado en el capítulo anterior, la fuente de nuestros datos es Garmin Connect, el problema que nos encontrábamos con esta fuente es que no podíamos automatizar un proceso que descargará datos de usuarios de la aplicación ya que, estos datos eran privados. Por este motivo la opción que elegimos para obtener estos datos fue crear un cuestionario online, en el que los deportistas después de contestar una serie de preguntas iniciaran sesión en la aplicación, con un formulario como el de la Figura 4.1.

Al no querer almacenar datos sensibles, como nombres de usuarios y contraseñas de los deportistas, decidímos que la mejor opción era descargar todas las actividades utilizando la API de Garminconnect una vez el usuario hubiera iniciado sesión en la aplicacion. El problema que nos encontrábamos ahora es que el servidor que elegimos para alojar nuestra aplicación, Heroku, borraba todos los ficheros descargados al poco tiempo de descargarlos, por lo que en vez de descargar las actividades y guardarlas en ficheros GPX, optamos por almacenarlas directamente en base de datos.

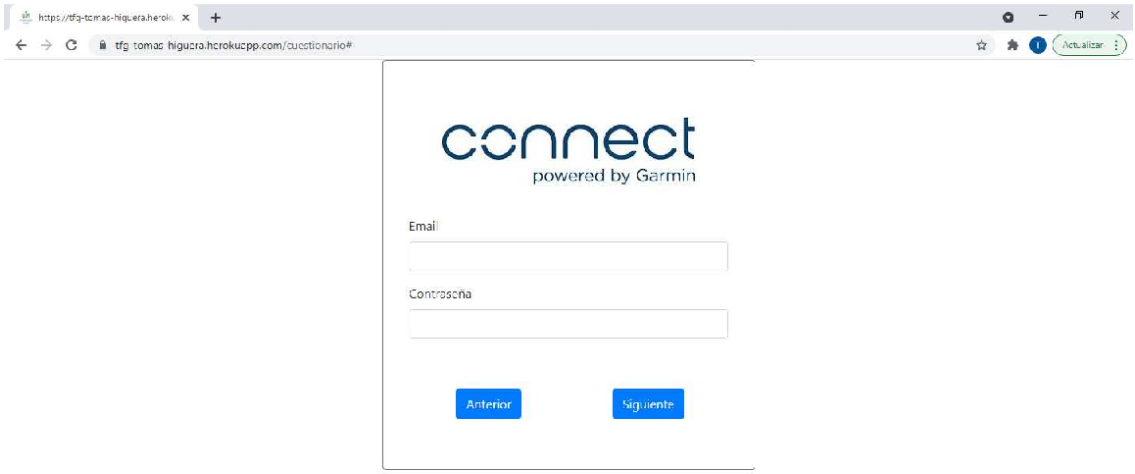

FIGURA 4.1: Inicio de sesión en Garminconnect para la descargar actividades.

4.2. Estructura de la base de datos

A lo largo de esta sección se definirá la estrucutura y el tipo de base de datos elegida en este proyecto. La base de datos elegida fue una de tipo relacional, PostgreSQL. Los diferentes motivos por los que elegimos esta base de datos son los siguientes:

- **Servidor de aplicación:** Al haber elegido Heroku como servidor de aplicación, PostgreSQL era una de las opciones más favorables para nosotros, ya que este servidor de aplicaciones permite utilizar esta base de datos de manera gratuita, con un espacio limitado, pero suficiente para nuestro proyecto.
- **Framework:** Al haber elegido Django como framework para nuestra aplicación, PostgreSQL era una buena opción para nostotros, ya que este framework es compatible con este tipo de base de datos y nos permite interactuar con ella muy fácilmente.

El siguiente paso despúes de elegir el tipo de base de datos que íbamos a utilizar fue definir qué información necesitábamos de los deportistas, a parte de los entrenamientos procedentes de Garminconnect. La necesidad de obtener información adicional a los entrenamientos era ayudarnos a detectar los patrones que buscábamos a lo largo de la vida deportiva de un deportista de resistencia. Para obtener esta información decidimos realizar un cuestionario en la aplicación web, para que los deportistas rellenaran antes de descargar sus datos. Este cuestionario no podía ser muy extenso ya que, sabíamos que la

gran mayoría de los usuarios que recibieran este cuestionario, no se pararían a rellenarlo si este le hiciera emplear mucho tiempo. Finalmente optamos por el cuestionario que aparece en la Figura 4.2

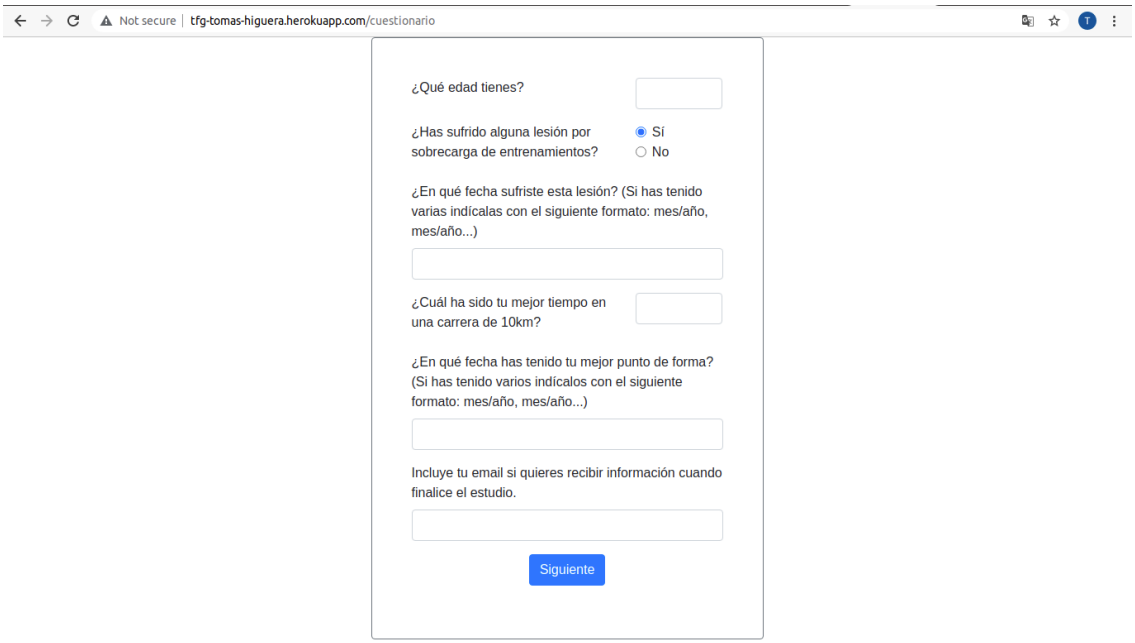

FIGURA 4.2: Cuestionario de la aplicación web.

Optamos por elegir estas preguntas ya que, tras hablar con varios entrenadores que utilizaban la unidad TSS para medir la carga de entrenamientos en sus deportistas, nos manifestaron que gracias a esta medida podríamos intentar predecir tanto las lesiones de un deportista, como el mejor momento para competir de un deportista. Con estas preguntas conoceríamos según la perspectiva del deportista, en qué momentos había estado lesionado y cuándo había estado en su mejor momento de forma para despúes, mediante un análisis visual de sus gráficas de CTL, ATL y TSB, buscar patrones que dieran lugar a estas situaciones.

4.3. Visualización y limpieza de la base de datos

Tras lanzar la aplicación web, obtuvimos una base de datos de 22 deportistas, con un total de 24274 actividades. Al no ser una gran cantidad de deportistas, decimos visualizar las gráficas de Fitness, Fatigue y Form en toda la vida deportiva de cada individuo.Gracias a esta primera visualización observamos un usuario cuyos resultados no eran coherentes. En la Figura 4.3 se puede observar como el Fitness del usuario es demasiado alto, superando cien en varias ocasiones, cuando lo normal en un deportista no profesional de nivel alto es estar entre cincuenta y setenta.

Para conocer el motivo de la obtención de estos resultados, decidimos hablar con el deportista que con el que habíamos obtenido estos datos. Este individuo nos trasladó que el altímetro de su GPS estaba roto, por lo que, al influir la altitud a la hora de normalizar

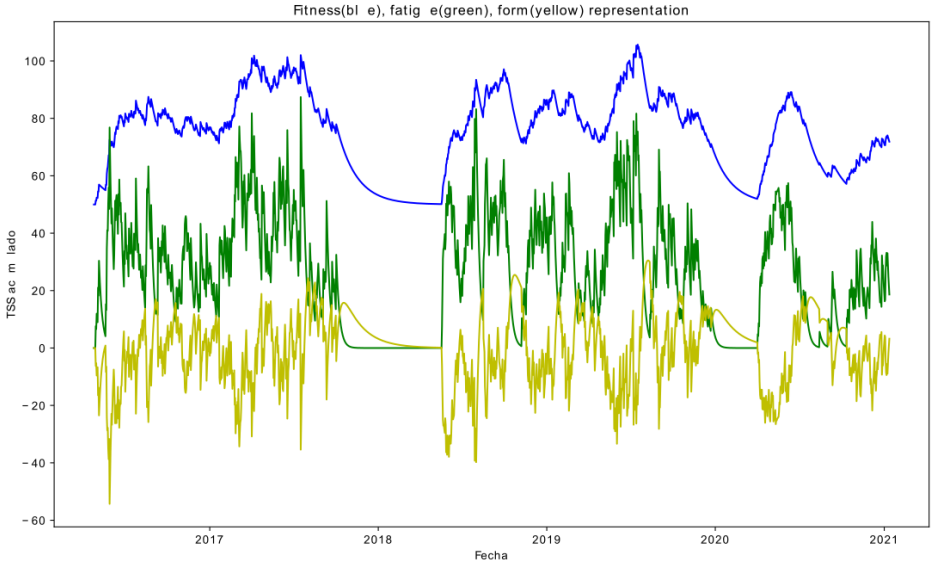

FIGURA 4.3: Usuario eliminado de la base de datos.

la velocidad de los entrenamientos, el cálculo de los TSS no iba a ser el correcto. Al no haber más usuarios como este y que el resto de los resultados parecían coherentes en esta primera visualización, decidimos eliminar a este usuario de la base de datos y dejar el resto para el estudio. Nuestra base de datos final contaba con 21 usuarios y 23602 entrenamientos.

4.4. Búsqueda de patrones y elección de atributos

Una vez creada la base de datos, pasamos a realizar una visualización mas exhaustiva de los datos, para buscar patrones en las fechas de mejor punto de forma y lesiones de los deportistas de nuestra base de datos.

4.4.1. Lesiones

La primera respuesta que analizamos fue la de las lesiones. Antes de visualizar los resultados obtenidos, hablamos con entrenadores para que nos dieran las pautas para tratar de detectar una lesión en un deportista de resistencia. El proceso que ellos seguían para detectar si un deportista estaba realizando una carga excesiva en sus entrenamientos, era observar la curva de Fitness, que se corresponde con la media ponderada exponencialmente de los TSS en los últimos cuarenta y dos días. Si esta curva tenía una pendiente elevada significaba que el deportista había acumulado mucha fatiga en los últimos días, por lo que podría sufrir una lesión por sobrecarga.

Esta sobrecarga tambíen se podía observar en las curvas Fatigue y Form. En la curva Fatigue, que se corresponde con la media ponderada exponencialmente de los TSS en los últimos siete días, si se producían picos positivos con una gran pendiente. En la curva Form, que es la diferencia entre Fitness y Fatigue del día anterior, si se producian picos negativos con una gran pendiente.

Una vez explicadas las pautas, pasamos a visualizar los datos junto a las respuestas. El problema que nos encontrábamos a la hora de reconocer este tipo de patrones, es que la mayor parte de deportistas no se había lesionado, por lo que no teníamos muchas gráficas en las que analizar porqué se hbían dado estas lesiones.

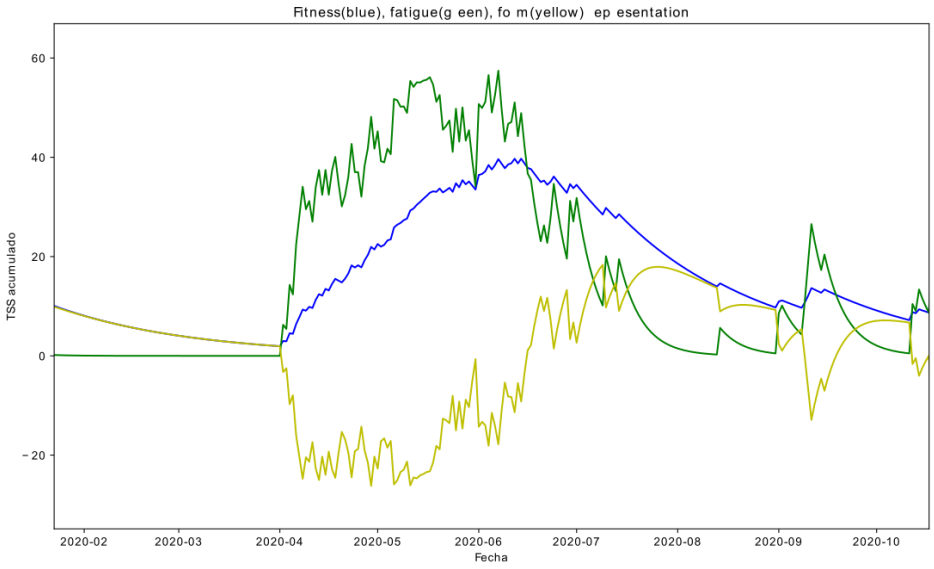

FIGURA 4.4: Deportista lesionado en Junio de 2020.

En la Figura 4.4 se muestra la vida deportiva de un deportista que se lesionó en Junio de 2020. En esta Figura podemos observar como el deportista previamente a la lesión presentaba esa pendiente elevada de Fitness (línea azul) de la que hablaban los entrenadores. A la hora de comparar con otros deportistas observábamos que no siempre este patrón de pendiente positiva en el Fitness era sinónimo de una lesión, como podemos observar en la Figura 4.5, en la que el deportista sufrió una lesión en Marzo de 2020.

Con estas visualizaciones, concluímos que con la poca cantidad de datos que teníamos no podíamos detectar estos patrones previos a una lesión, por lo que pasamos a la siguiente predicción.

4.4.2. Mejor punto de forma

Para detectar patrones de mejor punto de forma seguimos la misma metodología que con las lesiones. Primero hablamos con los entrenadores para que nos explicaran

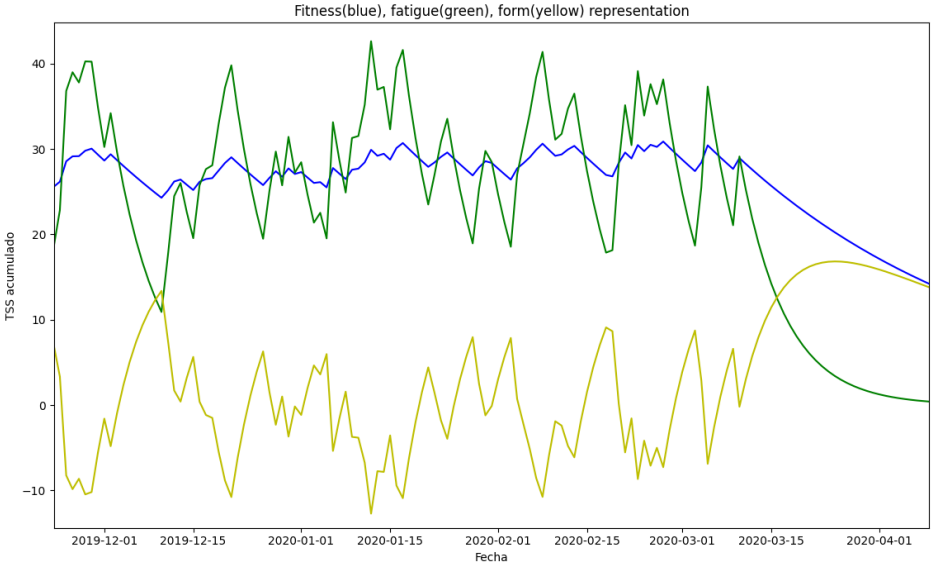

FIGURA 4.5: Deportista lesionado en Marzo de 2020.

como ellos ajustaban las cargas para tratar que sus deportistas llegaran en un punto óptimo de forma en las competiciones. Todos los entrenadores coincidían que esos picos de forma se obtenían cuando el Fitness del deportista era más alto y el valor de Form se situaba entre menos diez y diez. Una vez conocidas estas pautas pasamos a visualizar la vida deportiva de los deportistas de resistencia. En este caso todos los deportistas habían respondido a la pregunta de su mejor punto de forma, por lo que nos resultó más sencillo analizar los resultados.

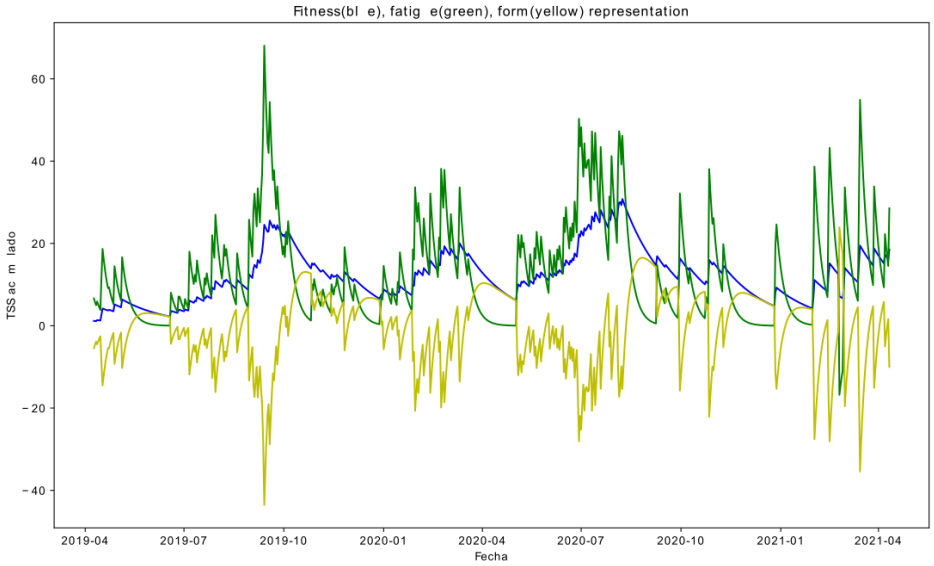

FIGURA 4.6: Deportista con mejor punto de forma en Septiembre de 2020.

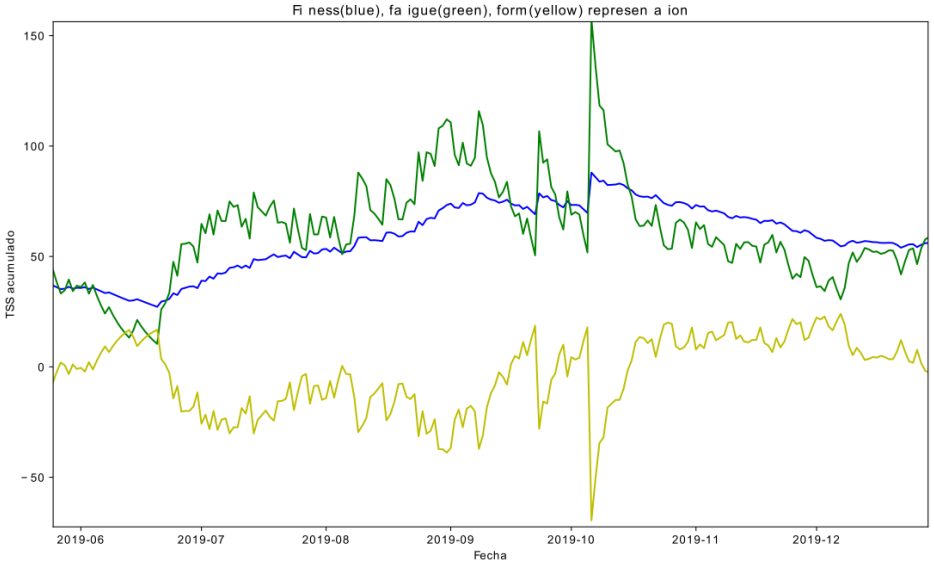

FIGURA 4.7: Deportista con mejor punto de forma en Septiembre de 2019.

Cómo se puede observar en las Figuras 4.6, 4.7, 4.8, las respuestas de los deportistas coincidían con lo expuesto por los entrenadores, un Fitness alto y unos valores de Form entre menos y diez y diez, por lo que los resultados obtenidos nos valdrían para el estudio.

4.4.3. Elección de atributos

Una vez elegido el patrón que ibamos a estudiar, pasamos a elegir los atributos que usaríamos en los algoritmos de aprendizaje automático [18]. Optamos por realizar una primera aproximación en la que guardaríamos los valores de Fitness, Fatigue y Form de los últimos treinta días. En esta primera aproximación no tendríamos en cuenta velocidades ni el TSS concreto de los entrenamientos. Finalmente mantuvimos estos atributos porque los resultados obtenidos al finalizar el estudio fueron positivos como se explica más adelante.

4.5. Extracción de puntos característicos

En esta sección se explicará la metodología aplicada para la extracción de puntos característicos. Para la aplicación de los algoritmos de aprendizaje automático era necesario extraer puntos positivos y puntos negativos con los que realizaríamos el entrenamiento. A continuación se explica como realizamos la extracción de los mismos.

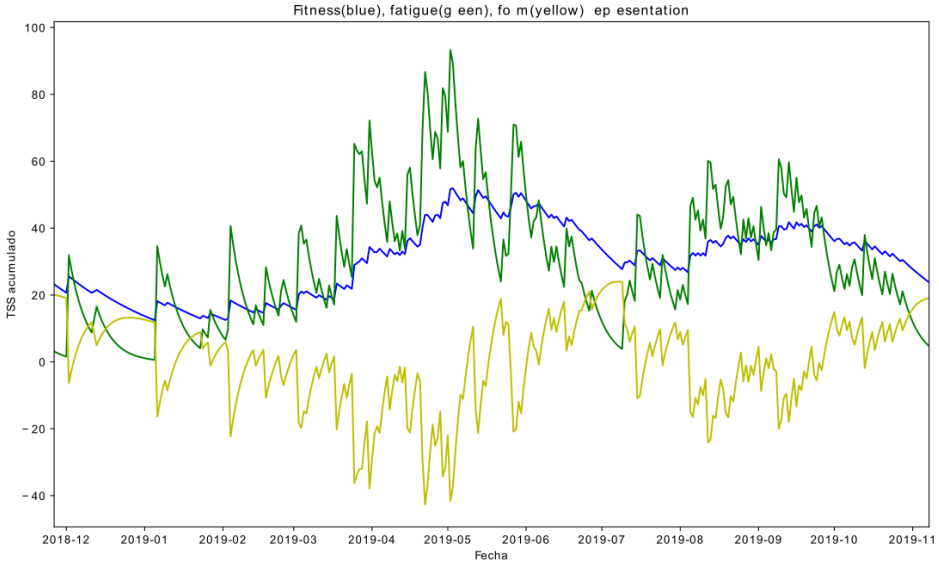

FIGURA 4.8: Deportista con mejor punto de forma en Junio de 2019.

4.5.1. Puntos positivos

Para la obtención de los puntos positivos para nuestro algoritmo de aprendizaje automático seguiremos la siguiente metodología:

1. Primero buscaremos los mejores puntos de forma. Estos puntos se corresponden con los máximos locales de Fitness a lo largo de la vida deportiva de un deportista y cuya forma se sitúe entre los valores menos diez y diez.

2. Una vez obtenidos los puntos extraeremos los valores correspondientes a Fitness, Fatigue y Form de los treinta días anteriores y los almacenaremos en un fichero de tipo csv.

Como podemos observar en la Figura 4.9, a lo largo de la vida deportiva de un deportista, el Fitness (línea azul) no tiene una subida constante, sino que realiza subidas y bajadas antes de llegar a los picos más altos. Por este motivo, si tratábamos de obtener máximos locales íbamos a obtener un gran cantidad de falsos positivos. Finalmente optamos por aplicar un filtro de Savitzky-Golay [19], que no serviría para suavizar los resultados y con ello obtener los máximos locales de la función de Fitness . En la Figura 4.9 se puede observar el resultado tras aplicar el suavizado con este filtro. La línea roja se corresponde con la función de Fitness suavizada.

Trás el procesado y extracción de puntos positivos de los deportistas, obtuvimos un total de 642 puntos positivos para nuestro modelo de aprendizaje automático.

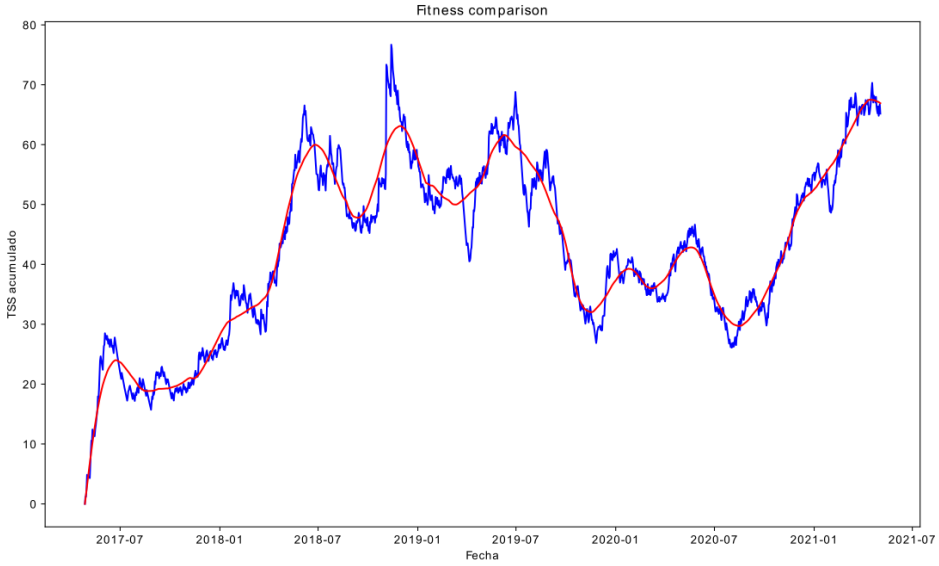

FIGURA 4.9: Grafica de Fitness con filtro de suavizado aplicado.

4.5.2. Puntos negativos

Trás obtener los puntos positivos pasamos a obtener puntos negativos para nuestro modelo de aprendizaje. Esta búsqueda sería más sencilla, ya que buscaríamos puntos aleatorios entre todos los deportistas, que no se correspondieran con los puntos positivos obtenidos previamente. Obtuvimos una cantidad de 1284 puntos, el doble respecto a los puntos positivos encontrados.

```
Usuario 4 -> Se han extraido 60 valores positivos de los cuales son validos 31
Usuario 5 -> Se han extraido 60 valores positivos de los cuales son validos 45
Usuario 8 -> Se han extraido 51 valores positivos de los cuales son validos 40
Usuario 9 -> Se han extraido 30 valores positivos de los cuales son validos 20
Usuario 11 -> Se han extraido 27 valores positivos de los cuales son validos 14
Usuario 12 -> Se han extraido 39 valores positivos de los cuales son validos 12
Usuario 13 -> Se han extraido 63 valores positivos de los cuales son validos 15
Usuario 14 -> Se han extraido 54 valores positivos de los cuales son validos 31
Usuario 16 -> Se han extraido 135 valores positivos de los cuales son validos 67
Usuario 17 -> Se han extraido 75 valores positivos de los cuales son validos 39
Usuario 19 -> Se han extraido 78 valores positivos de los cuales son validos 34
Usuario 20 -> Se han extraido 75 valores positivos de los cuales son validos 31
Usuario 23 -> Se han extraido 51 valores positivos de los cuales son validos 19
Usuario 24 -> Se han extraido 21 valores positivos de los cuales son validos 13
Usuario 28 -> Se han extraido 87 valores positivos de los cuales son validos 37
Usuario 29 -> Se han extraido 93 valores positivos de los cuales son validos 76
Usuario 31 -> Se han extraido 21 valores positivos de los cuales son validos 9
Usuario 42 -> Se han extraido 39 valores positivos de los cuales son validos 36
Usuario 44 -> Se han extraido 75 valores positivos de los cuales son validos 54
Usuario 0 -> Se han extraido 39 valores positivos de los cuales son validos 19
Se han extraido un total de 642 puntos positivos
```

FIGURA 4.10: Puntos positivos obtenidos.

```
Se han extraido un total de 642 puntos positivos
Se van a buscar 1284 puntos negativos
Se han extraido un total de 1284 puntos negativos
Se han extraido un total de 1926 puntos
```

FIGURA 4.11: Puntos negativos obtenidos.

Capítulo 5

Pruebas y resultados

En este capítulo se describirá el análisis de los resultados obtenidos tras completar el desarrollo del modelo de aprendizaje automático propuesto en este trabajo. Este análisis contendrá explicaciones acerca del modelo de aprendizaje elegido, así como la precisión obtenida trás el entrenamiento.

5.1. Diseño de los experimentos de aprendizaje automático

En esta sección se describirán los diferentes experimentos realizados, el modelo de aprendizaje automático elegido y la distribución de los datos. Una vez obtenidos los puntos característicos y guardados en un fichero csv, pasamos a crear nuestro modelo de aprendizaje automático. El modelo final elegido es Random Forest. El motivo de elegir este modelo es los buenos resultados que da y que es capaz de manejar un gran cantidad de variables de entrada e identificar las más significativas. Para contrastar los resultados de este modelo y obtener mejores predicciones también realizamos pruebas con el modelo SVM, otro algoritmo de aprendizaje supervisado.

Para la distribución de los datos para entrenamiento y clasificación optamos por la validación cruzada. El motivo de elegir validación cruzada fue que el rendimiento que se obtiene con este tipo de distribución con conjuntos de datos pequeños es mejor que con validación simple.

Para la realización de estos experimentos hicimos uso de los módulos de Python scikit-learn [20] para el modelo y la distribución de los datos. También hicimos uso del módulo pandas [21] y numpy [22] para la lectura de los datos del archivo csv.

El primer experimento realizado fue un modelo Random Forest con validación cruzada. La configuración del modelo Random Forest es la estándar, ya que antes de empezar a cambiar parámetros en la configuración queríamos comprobar si los resultados con la configuración por defecto eran positivos.

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold

# Creacion modelo random forest
model = RandomForestClassifier()
# Aplico validacion cruzada con el modelo creado
scores = cross_val_score(model, X=datos, y=clases, cv=
    StratifiedKFold(n_splits=10, shuffle=True))
```

El siguiente experimento a realizar fue con un modelo SVM con validación cruzada. Al igual que con Random Forest, la primera prueba que realizaríamos sería con la configuración por defecto para comprobar que resultados obteníamos.

```
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

# Creacion modelo SVC
model = SVC()
# Aplico validacion cruzada con el modelo creado
scores = cross_val_score(model, X=datos, y=clases, cv=
    StratifiedKFold(n_splits=10, shuffle=True))
```

5.2. Análisis de los resultados

En esta sección se analizarán los resultados obtenidos en los experimentos descritos anteriormente. Para analizar los resultados obtenidos hemos optado por usar como medida la precisión y la desviación estándar del modelo trás aplicar *cross_val_score*.

```
(TFG) p3arl@p3arl-TM1613:~/Documents/TFG$ python randomForest.py
Aplicando random forest con validacion cruzada...
Resultados obtenidos...
Accuracy: 0.92 (+/- 0.04)
```

FIGURA 5.1: Rendimiento Random Forest.

5.2.1. Puntos negativos

Como podemos observar en la Figura 5.1 el rendimiento obtenido al aplicar validación cruzada con un modelo Random Forest con la configuración por defecto de sklearn fue del 92 %.

```
Aplicando SVC con validacion cruzada...
Resultados obtenidos...
Accuracy: 0.67 (+/- 0.01)
```

FIGURA 5.2: Rendimiento SVM.

Como podemos observar en la Figura 5.2 el rendimiento obtenido al aplicar validación cruzada con un modelo SVM con la configuración por defecto de sklearn fue del 67 %.

Los resultados obtenidos con Random Forest son ligeramente mejores que los de SVC. Podemos concluir que el estudio ha sido satisfactorio, ya que la precisión obtenida ha sido de más del 92 %, un buen resultado teniendo en cuenta la cantidad de datos de la que disponíamos.

Capítulo 6

Conclusiones y trabajo futuro

Para terminar con esta memoria, en este capítulo se expondrán las conclusiones a las que se ha llegado tras la realización del trabajo y las ideas para mejorar este estudio en el futuro.

6.1. Conclusiones

Para concluir, podemos confirmar que este proyecto ha cumplido con los objetivos planteados al comienzo del mismo. El aprendizaje automático aplicado al deporte, es todavía un nicho sin explorar y es por ello que este proyecto partía de ser un estudio con pocos antecedentes, por lo que no sabíamos la viabilidad del mismo. Finalmente hemos sido capaces de desarrollar un modelo de aprendizaje automático que es capaz de procesar todos los entrenamientos de un deportista y predecir su mejor punto de forma para competir.

Esta herramienta es de gran utilidad para deportistas que quieran batir sus mejores marcas, ya que viendo su progresión a lo largo del tiempo, nuestro modelo de aprendizaje es capaz de obtener una visión objetiva de la forma física del atleta y con ello realizar una predicción. Esta herramienta puede también favorecer el trabajo de los entrendores, dándoles una métrica más que refleje el estado de forma de sus deportistas.

Para finalizar, a modo de reflexión personal, este trabajo ha supuesto un verdadero reto, y no solo a nivel infórmatico. Para completar este proyecto ha sido necesario hablar con entrenadores y deportistas, que nos acercasen un poco más a su día a día de cara a los entrenamientos y que nos explicarán las técnicas que aplicaban al deporte de resistencia. Para el resto del desarrollo del proyecto ha sido necesario aplicar los conocimientos adquiridos durante estos años, así como llevar una planificación que nos permitiese acabar el proyecto dentro de los plazos permitidos.

6.2. Trabajo futuro

A lo largo del proyecto se han planteado infinidad de mejoras que se podrían realizar en trabajos futuros. Gran parte de estas mejoras no se han podido llevar a cabo por el escaso tamaño de nuestra base de datos. A continuación se expondrán algunas de las planteadas:

- Tratar de predecir lesiones en deportistas. A lo largo de este proyecto hemos tratado de realizar predicciones por sobrecarga de entrenamientos. Esta tarea no ha podido ser llevada a cabo por la pequeña cantidad de datos que disponíamos, ya que teníamos pocos ejemplos prácticos que visualizar para detectar este tipo de patrones.
- Tratar de enfocar esta herramienta a la planificación de entrenamientos que realizan los entrenadores. Al comienzo de este proyecto planteabamos la posibilidad de haciendo uso del aprendizaje automático ayudar a los entrenadores a planificar entrenamientos para sus deportistas. El modo en el que planteabamos ayudarles era dar un feedback acerca de como se encontraba el deportista, si podía asimilar más carga de entrenamiento, etc. Finalmente, debido a la magnitud de este proyecto, no pudimos abordar este tema.
- Otra posible mejora sería optimizar los cálculos de TSS, ya que en este proyecto se realizan muchas estimaciones. Se podrían realizar estadísticas con una gran base de datos de entrenamientos, para poder mejorar cálculos como el del ritmo normalizado. Además se podría integrar el uso de potenciómetros de carrera y de bicicleta para mejorar los cálculos en la fórmula para obtener los TSS.

Bibliografía

[1] Ministerior de cultura y deporte. Gobierno de españa. *Anuario de estadísticas deportivas 2020*. 2020. URL: https://www.culturaydeporte.gob.es/dam/jcr:47414879-4f95-4cae-80c4-e289b3fbced9/anuario-de-estadisticas-deportivas-2020.pdf (visitado 16-06-2020).

[2] Django. *Django*. 2020. URL: https://docs.djangoproject.com/en/3.2/ (visitado 16-06-2020).

[3] Yanette Díaz González y Yenisleidy Fernández Romero. «Patrón Modelo-Vista-Controlador.» En: *Telemática* 11.1 (jun. de 2012), págs. 47-57. URL: https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15.

[4] Bootstrap. *Bootstrap*. 2020. URL: https://getbootstrap.com/docs/4.1/about/overview/ (visitado 16-06-2020).

[5] PostgreSQL. *PostgreSQL*. 2020. URL: https://www.postgresql.org/about/ (visitado 16-06-2020).

[6] Heroku. *Heroku*. 2020. URL: https://www.heroku.com/about (visitado 16-06-2020).

[7] Nelly Lisbeth Hernandez y Anderson Smith Florez-Fuentes. «COMPUTACIÓN EN LA NUBE». En: *Mundo FESC* 4.8 (dic. de 2014), págs. 46-51. URL: https://www.fesc.edu.co/Revistas/OJS/index.php/mundofesc/article/view/48.

[8] Antonio Moreno. *Aprendizaje automático*. 1994.

[9] analyticsVidhya.com. *Random forest*. 2020. URL: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/ (visitado 16-06-2020).

[10] Garmin. *Garmin*. 2020. URL: https://connect.garmin.com/ (visitado 16-06-2020).

[11] Training Peaks. *Training Peaks*. 2020. URL: https://www.trainingpeaks.com/about-us/ (visitado 16-06-2020).

[12] Joe Friel. *The cyclist's training bible*. VeloPress, 2012.

[13] petergardfjall. *Garminexport*. 2020. URL: https://github.com/petergardfjall/garminexport (visitado 16-06-2020).

[14] Python. *XML*. 2020. URL: https://docs.python.org/3/library/xml.etree.elementtree.html (visitado 16-06-2020).

[15] C Carl Robusto. «The cosine-haversine formula». En: *The American Mathematical Monthly* 64.1 (1957), págs. 38-40.

[16] tristarathletes.com. *Ritmo normalizado*. 2020. URL: https://www.tristarathletes.com/coach-blog-and-bios/treadmillchart (visitado 16-06-2020).

[17] Matplotlib. *Matplotlib*. 2020. URL: https://matplotlib.org/stable/users/history.html (visitado 16-06-2020).

[18] Pedro Pury. «Fundamentos de Aprendizaje Automático». En: *Córdoba, España. Recuperado de http://www. famaf. proed. unc. edu. ar/pluginfile. php/19002/mod_resource/content/2/01. introduccion. pdf* (2015).

[19] Ronald W Schafer. «What is a Savitzky-Golay filter?[lecture notes]». En: *IEEE Signal processing magazine* 28.4 (2011), págs. 111-117.

[20] Scikit-learn. *Scikit-learn*. 2020. URL: https://scikit-learn.org/stable/ (visitado 16-06-2020).

[21] Pandas. *Pandas*. 2020. URL: https://pandas.pydata.org/ (visitado 16-06-2020).

[22] Numpy. *Numpy*. 2020. URL: https://numpy.org/ (visitado 16-06-2020).

[23] PasoClave.com. *Modelo ATR*. 2020. URL: https://www.pasoclave.com/atr-periodizacion-entrenamiento-escalada/ (visitado 16-06-2020).

Anexos

Anexo A

Metodologías de entrenamiento

En la actualidad, los entrenadores utilizan el model ATR para la planificación de entrenamientos de sus deportistas. El objetivo de este modelo es buscar objetivos a medio-corto plazo y preparar al deportista para los mismos, consiguiendo que este no pierda la motivación.

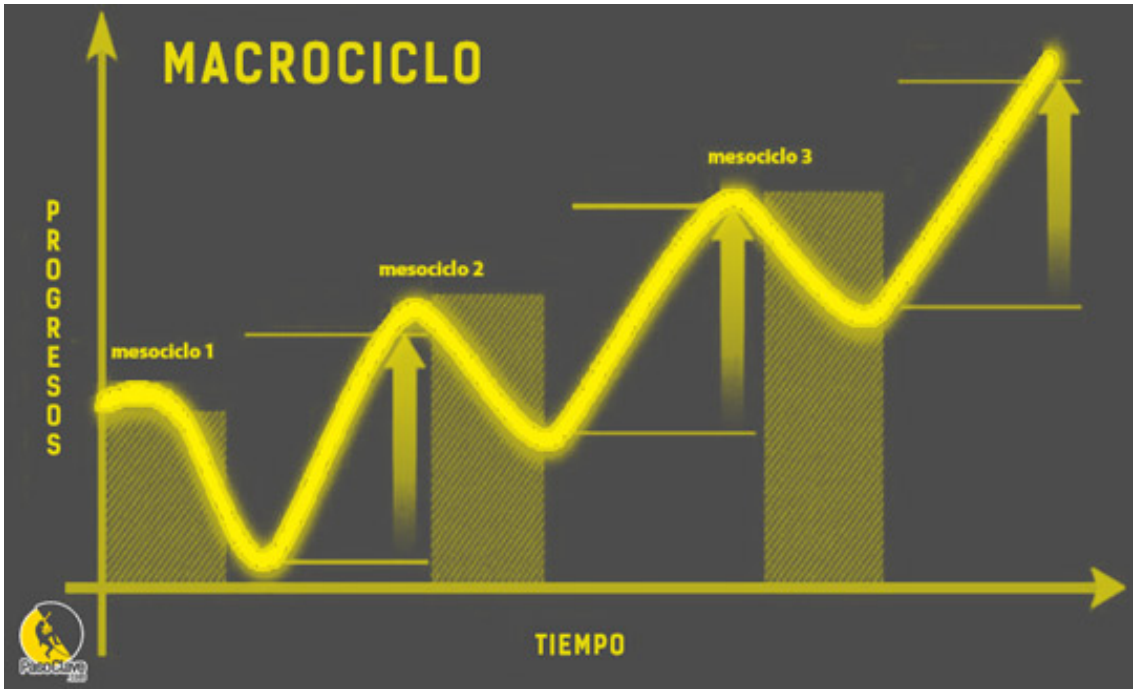

FIGURA A.1: Modelo ATR. Imagen extraída de [23].

Una planificación de una temporada de entrenamiento se corresponde con un macrociclo. Dentro de un macrociclo hay mesociclos y dentro de estos hay microciclos que corresponden con una semana de entrenamiento. Dependiendo del nivel del deportista se siguen unas pautas para planificar los entrenamientos, en deportistas amteur se sigue la siguiente metodología:

1. Dos o tres microciclos de carga.
2. Un microciclo de descarga.

3. Si se trata de una semana de competición, un microciclo de tappering.

A la hora de planificar entrenamientos también hay que tener en cuenta dos factores, el volumen y la intensidad. A medida que el deportista se acerca a la competición se disminuye el volumen en los entrenamientos y se aumenta la intesidad para poder llegar en el mejor momento de forma.

Una buena medida para medir la intensidad en los entrenamientos es la frecuencia cardíaca. Para calcular las frecuencias cardíacas de trabajo de un deportista se debe realizar una prueba de esfuerzo que indique el LTHR (frecuencia cardiaca umbral láctica). Si el deportista no ha realizado ninguna prueba de esfuerzo se puede estimar la frecuencia cardiaca máxima con la siguiente fórmula:

$$FCmax = 220 - edad$$

Las zonas de trabajo son las siguientes:

- **Entrenamiento aeróbico**
  - Zona 1 (Recovery): Para calentamiento y recuperación entre series.
    <85 % LTHR
    50-60 % FCmax
  - Zona 2 (Aerobic): Para quema de grasas (Entrenamientos de veinte minutos o más).
    85-89 % LTHR
    60-70 % FCmax
  - Zona 3 (Tempo): Para mejorar la resistencia (Intervalos de cinco a veinte minutos con recuperaciones de un minuto por cada cinco).
    90-94 % LTHR
    60-70 % FCmax
  - Zona 4 (SubThreshHold): Para retrasar la aparición de lactato (Intervalos de cinco a veinte minutos con recuperaciones de un minuto por cada cinco).
    95-99 % LTHR
    70-80 % FCmax
- **Entrenamiento aneróbico**
  - Zona 5A (SuperThreshHold): Para retrasar la aparición de lactato (Intervalos de cinco a veinte minutos con recuperaciones de un minuto por cada cinco).
    100-102 % LTHR
    70-80 % FCmax

- Zona 5B (Aerobic capacity): Para mejorar el V02max (Intervalos de tres a siete minutos con recuperaciones similares a los intervalos).
  95-99 % LTHR
  70-80 % FCmax
- Zona 5C (Anaerobic capacity): Para mejorar la velocidad máxima (Intervalos de dos minutos máximo con recuperaciones al menos de la duración del intervalo).
  95-99 % LTHR
  70-80 % FCmax